Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 19

Студент: Керимов А. Ш.

Группа: ИУ7-64Б

Преподаватель: Толпинская Н. Б.

Москва.
2020 г.

Цель работы – изучить способы организации, представления и обработки списков в программах на Prolog, методы создания эффективных рекурсивных программ обработки списков и порядок их реализации.

Задание.

Ответить на вопросы (коротко**):**

1. Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как можно организовать выход из рекурсии в Prolog?
2. Какое первое состояние резольвенты?
3. В каких пределах программы переменные уникальны?
4. В какой момент, и каким способом системе удается получить доступ к голове списка?
5. Каково назначение использования алгоритма унификации?
6. Каков результат работы алгоритма унификации?
7. Как формируется новое состояние резольвенты?
8. Как применяется подстановка, полученная с помощью алгоритма унификации – как глубоко?
9. В каких случаях запускается механизм отката?
10. Когда останавливается работа системы? Как это определяется на формальном уровне?

Используя хвостовую рекурсию, разработать эффективную программу, (комментируя назначение аргументов), позволяющую:

1. Найти длину списка (по верхнему уровню);
2. Найти сумму элементов числового списка
3. Найти сумму элементов числового списка, стоящих на нечётных позициях исходного списка (нумерация от 0)

Убедиться в правильности результатов

Для одного из вариантов **ВОПРОСА** и одного из **заданий составить таблицу**, отражающую конкретный порядок работы системы:

Т.к. резольвента хранится в виде стека, то состояние резольвенты требуется отображать в столбик: вершина – сверху! Новый шаг надо начинать с нового состояния резольвенты! Для каждого запуска алгоритма унификации, требуется указать № выбранного правила и дальнейшие действия – и почему.

Текст процедуры, Вопрос:…..

№ шага	Текущая резольвента – ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия с комментариями
шаг1	…	…	…
…	…	…	…

Практическая часть

Листинг 1. Факториал и Фибоначчи

```
domains
  elements = integer*

predicates
  length(elements List, integer Len, integer CurLen)
  sum(elements List, integer Sum, integer CurSum)
  sumOdd(elements List, integer Sum, integer CurSum)

clauses
  % длина списка
  length([], Len, Len) :- !.
  length([_|T], Len, PrevLen) :- CurLen = PrevLen + 1, length(T, Len, CurLen).

  % сумма элементов списка
  sum([], Sum, Sum) :- !.
  sum([H|T], Sum, PrevSum) :- CurSum = PrevSum + H, sum(T, Sum, CurSum).

  % сумма элементов списка на нечётных позициях
  sumOdd([], Sum, Sum) :- !.
  sumOdd([_|[]], Sum, Sum) :- !.
  sumOdd([_, N|T], Sum, PrevSum) :- CurSum = PrevSum + N, sumOdd(T, Sum, CurSum).

goal
  length([], Len, 0).                    % Len = 0
  %length([1], Len, 0).                  % Len = 1
  %length([1, 2, 3], Len, 0).            % Len = 3
  %sum([], Sum, 0).                      % Sum = 0
  %sum([1], Sum, 0).                     % Sum = 1
  %sum([1, 2, 3], Sum, 0).               % Sum = 6
  %sumOdd([], Sum, 0).                   % Sum = 0
  %sumOdd([1], Sum, 0).                  % Sum = 0
  %sumOdd([1, 2, 3, 4, 5, 6], Sum, 0). % Sum = 12
```

Таблица для цели sum([1, 2, 3], Sum, 0).

№ шага	Текущая резольвента – ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия с комментариями
1	sum([1, 2, 3], Sum, 0)	ТЦ: sum([1, 2, 3], Sum, 0). Поиск с начала базы знаний. Правило I: length([], Len1, Len1). ПРI: sum(…); length(…). Неудача, разные функторы. Возврат к ТЦ, метка переносится ниже. Правило II: length([_|T1], Len1, PrevLen1). ПРII: sum(…); length(…). Неудача, разные функторы. Возврат к ТЦ, метка переносится ниже. Правило III: sum([], Sum1, Sum1). ПРIII: [1, 2, 3] = []. Неудача при сопоставлении списков. Возврат к ТЦ, метка переносится ниже.	Проверка тела ПРIV. Изменение резольвенты в 2 этапа.

		Правило IV: sum([H1|T1], Sum1, PrevSum1). ПРIV: [1, 2, 3] = [H1|T1]. Успех, подстановка {H1=1, T1=[2, 3], Sum=Sum1, PrevSum1=0}.	
2	CurSum1 = 0 + 1 sum([2, 3], Sum1, CurSum1)	ТЦ: CurSum1 = 0 + 1 Знак «=» означает конкретизацию свободной переменной CurSum1. {CurSum1 = 1}	Изменение резольвенты в 2 этапа.
3	sum([2, 3], Sum1, 1)	ТЦ: sum([2, 3], Sum1, 1). Поиск с начала базы знаний. Правило I: length([], Len2, Len2). ПРI: sum(…); length(…). Неудача, разные функторы. Возврат к ТЦ, метка переносится ниже. Правило II: length([_|T2], Len2, PrevLen2). ПРII: sum(…); length(…). Неудача, разные функторы. Возврат к ТЦ, метка переносится ниже. Правило III: sum([], Sum2, Sum2). ПРIII: [2, 3] = []. Неудача при сопоставлении списков. Возврат к ТЦ, метка переносится ниже. Правило IV: sum([H2|T2], Sum2, PrevSum2). ПРIV: [2, 3] = [H2|T2]. Успех, подстановка {H2=2, T2=[3], Sum1=Sum2, PrevSum2=1}.	Проверка тела ПРIV. Изменение резольвенты в 2 этапа.
4	CurSum2 = 1 + 2 sum([3], Sum2, CurSum2)	ТЦ: CurSum2 = 1 + 2 Знак «=» означает конкретизацию свободной переменной CurSum2. {CurSum2 = 3}	Изменение резольвенты в 2 этапа.
5	sum([3], Sum2, 3)	ТЦ: sum([3], Sum2, 3). Поиск с начала базы знаний. Правило I: length([], Len3, Len3). ПРI: sum(…); length(…). Неудача, разные функторы. Возврат к ТЦ, метка переносится ниже. Правило II: length([_|T3], Len3, PrevLen3). ПРII: sum(…); length(…). Неудача, разные функторы. Возврат к ТЦ, метка переносится ниже. Правило III: sum([], Sum3, Sum3). ПРIII: [3] = []. Неудача при сопоставлении списков. Возврат к ТЦ, метка переносится ниже.	Проверка тела ПРIV. Изменение резольвенты в 2 этапа.

		Правило IV: sum([H3|T3], Sum3, PrevSum3). ПРIV: [3] = [H3|T3]. Успех, подстановка {H3=3, T3=[], Sum2=Sum3, PrevSum3=3}.	
6	CurSum3 = 3 + 3 sum([], Sum3, CurSum3)	ТЦ: CurSum3 = 3 + 3 Знак «=» означает конкретизацию свободной переменной CurSum3. {CurSum3 = 6}	Изменение резольвенты в 2 этапа.
7	sum([], Sum3, 6)	ТЦ: sum([], Sum3, 6). Поиск с начала базы знаний. Правило I: length([], Len4, Len4). ПРI: sum(…); length(…). Неудача, разные функторы. Возврат к ТЦ, метка переносится ниже. Правило II: length([_|T4], Len4, PrevLen4). ПРII: sum(…); length(…). Неудача, разные функторы. Возврат к ТЦ, метка переносится ниже. Правило III: sum([], Sum4, Sum4). ПРIII: [] = []. Успех (подобрано знание). Подстановка {Sum4=6, Sum3=Sum4}.	Проверка тела ПРIII. Изменение резольвенты в 2 этапа.
8	!	ТЦ: !	Вывод: Sum=6 Завершение работы вследствие отсечения.

Теоретическая часть

1. Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как можно организовать выход из рекурсии в Prolog?

Рекурсия – один из способов организации повторных вычислений. В логическом программировании – способ заставить систему многократно использовать одну и ту же процедуру. При этом из неё должен быть выход.

Организация хвостовой рекурсии:

- Рекурсивный вызов единственен и расположен в конце тела правила.
- До вычисления рекурсивного вызова не должно быть возможности сделать откат (т. е. точки отката отсутствуют). Этого можно добиться, например, с помощью предиката отсечения.

Использовать отдельное правило, в конце которого будет находиться предикат отсечения.

2. Какое первое состояние резольвенты?

Если задан простой вопрос, то сначала он попадает в резольвенту.

Если вопрос представляет собой конъюнкцию нескольких термов, то резольвента будет содержать все эти термы, имея на вершине первый терм.

3. В каких пределах программы переменные уникальны?

Именованные переменные уникальны в пределах одного предложения, анонимные уникальны все.

4. В какой момент, и каким способом системе удаётся получить доступ к голове списка?

В Prolog существует более общий способ доступа к элементам списка. Для этого используется метод разбиения списка на начало и остаток. Начало списка – это группа первых элементов, не менее одного. Остаток списка – обязательно список (может быть пустой). Для разделения списка на начало, и остаток используется вертикальная черта (|) за последним элементом начала.
Если начало состоит из одного элемента, то получим: голову и хвост.

Во время унификации системе удаётся получить доступ к голове списка. В этот момент система пытается разделить список на «начало» и «конец», чтобы унификация была успешна.

5. Каково назначение использования алгоритма унификации?

Для поиска ответа на вопрос система должна найти подходящее знание.
Знание зафиксировано в заголовке правила.
Назначение алгоритма унификации — подобрать подходящее правило (подходящий заголовок).

6. Каков результат работы алгоритма унификации?

Унификация может завершаться успехом или тупиковой ситуацией (неудачей).

7. Как формируется новое состояние резольвенты?

Изменение резольвенты происходит в 2 этапа:
1) из стека выбирается подцель (верхняя, т.к. стек) и для неё выполняется редукция, т.е. замена подцели на тело найденного правила;
2) к полученной конъюнкции целей применяется подстановка (наибольший общий унификатор выбранной цели и заголовка сопоставленного с этой целью правила).

8. Как применяется подстановка, полученная с помощью алгоритма унификации – как глубоко?

Применение подстановки $\{X1=T1, \ldots, Xn=Tn\}$ заключается в замене каждого вхождения переменной Xi на соответствующий терм Ti. В результате подстановки конкретизированные переменные могут быть использованы для дальнейшего доказательства истинности тела правила.

9. В каких случаях запускается механизм отката?

Механизм отката запускается, если возникла тупиковая ситуация (достигнут конец БЗ) либо резольвента пуста. В таких случаях происходит откат к предыдущему состоянию резольвенты.

10. Когда останавливается работа системы? Как это определяется на формальном уровне?

Если достигнут конец базы знаний и нет альтернативных путей сопоставления (все метки выбранных ранее правил достигли конца БЗ), то работа завершается.

Исправления

№15

Унификация каких термов запускается на самом первом шаге работы… … унификация вопроса и первого предложения базы НЕТ – разные по структуре!!

Унификация заголовка вопроса и заголовка первого правила из БЗ.

№16

В каком случае система запускает алгоритм унификации? (Как эту необходимость на формальном уровне распознает система?)

Пролог выполняет унификацию в двух случаях: когда цель сопоставляется с заголовком предложения СИСТЕМА это знает заранее? или когда используется знак равенства, который является инфиксным предикатом ТАКОГО нет – это инфиксная форма записи, приближенная к общепринятой математической, а терм: =(T1, T2) (предикатом, который расположен между своими аргументами, а не перед ними).

Заранее система ничего не знает.
Если есть что доказывать (цель), то процесс унификации запускается автоматически.
Формально: если резольвента не пуста — запускается алгоритм унификации.
Процесс унификации можно запустить принудительно с помощью утверждения T1 = T2.

№17

Каково назначение использования алгоритма унификации? Алгоритм унификации необходим для попытки "увидеть одинаковость" – сопоставимость двух термов ЗАЧЕМ?

Для поиска ответа на вопрос система должна найти подходящее знание.
Знание зафиксировано в заголовке правила.
Назначение алгоритма унификации — подобрать подходящее правило (подходящий заголовок).